


SYSTEM FOR DYNAMIC LOAD OF SHARED LIBRARY AND METHOD THEREFOR

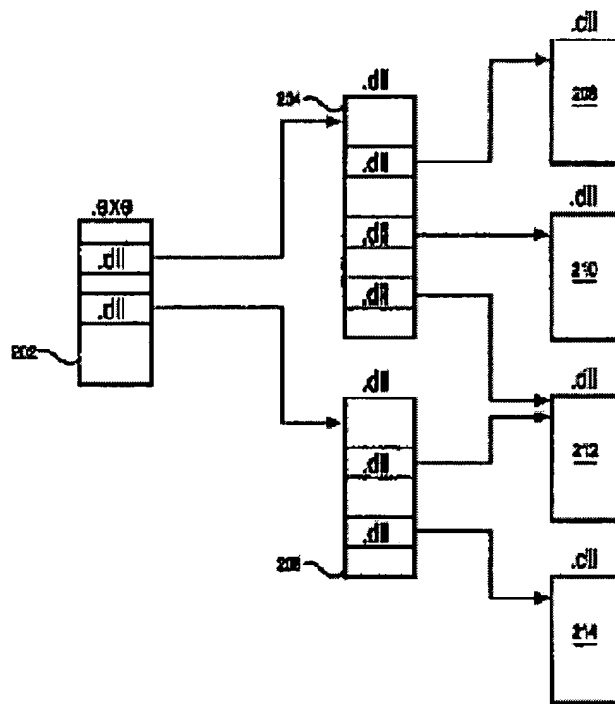
Patent number: JP6250924
Publication date: 1994-09-09
Inventor: ARENDT JAMES WENDELL; GIANGARRA PAUL
PLACIDO; MANIKUNDALAM RAVINDRANATH KASI;
PADGETT DONALD ROBERT; PARLAN JAMES
MICHAEL
Applicant: INTERNATL BUSINESS MACH CORP < IBM >
Classification:
- international: G06F12/08
- european:
Application number: JP19940005728 19940124
Priority number(s):

Also published as:

 EP0613084 (A1)[Report a data error here](#)**Abstract of JP6250924**

PURPOSE: To provide a mechanism for dynamically loading executable library objects only when the execution of those libraries is requested.

CONSTITUTION: In the dynamic load of executable library objects, the overhead of an operating system and necessary memory areas can be reduced by delaying the load of the objects until object reference is requested. In initial task loading, only main execution and the library objects referred to by the execution are assigned, and secondary reference objects are not assigned. The object reference generates page absence for a page to which the objects are assigned and not loaded. A page absence processing generates the loading of the executable objects and address rearrangement. A shared memory system permits the sharing of the executable objects until the objects are specifically referred to.



Data supplied from the esp@cenet database - Patent Abstracts of Japan

(51)Int.Cl.⁵

G 0 6 F 12/08

識別記号

庁内整理番号

Y 7608-5B

F I

技術表示箇所

審査請求 有 発明の数 9 OL (全 11 頁)

(21)出願番号 特願平6-5728

(22)出願日 平成6年(1994)1月24日

(31)優先権主張番号 0 2 3 6 4 3

(32)優先日 1993年2月26日

(33)優先権主張国 米国 (U S)

(71)出願人 390009531

インターナショナル・ビジネス・マシー
ズ・コーポレーションINTERNATIONAL BUSIN
ESS MASCHINES CORPO
RATIONアメリカ合衆国10504、ニューヨーク州
アーモンク (番地なし)

(72)発明者 ジュームス・ウェンデル・アレンド

アメリカ合衆国78681、テキサス州ラウン
ド・ロック、ブラックジャック・ドライブ
1501

(74)代理人 弁理士 合田 潔 (外2名)

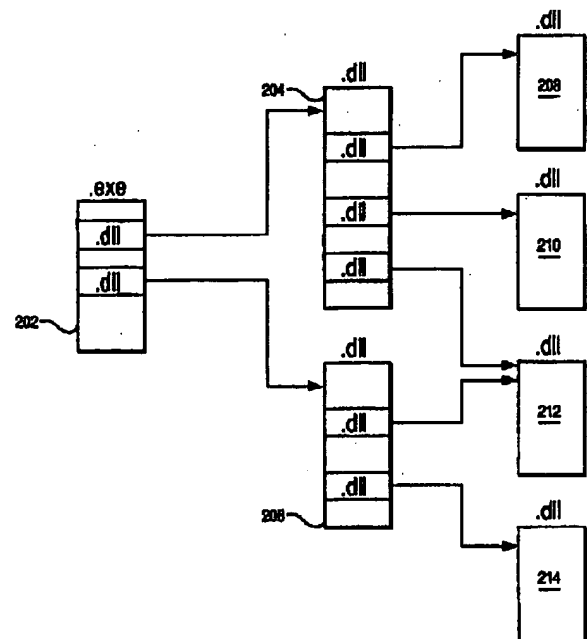
最終頁に続く

(54)【発明の名称】 共用ライブラリの動的ロードのためのシステム及び方法

(57)【要約】

【目的】 実行可能ライブラリ・オブジェクトが実行要求される時にだけ、これらのオブジェクトを動的にロードする機構を提供する。

【構成】 実行可能ライブラリ・オブジェクトの動的ロードが、オブジェクト参照が要求されるまでオブジェクトのロードを延期することにより、オペレーティング・システムのオーバーヘッド及び必要なメモリ領域を低減する。初期のタスク・ローディングはメイン実行及びその実行により参照されるライブラリ・オブジェクトだけを割当て、2次参照オブジェクトは割当てられない。オブジェクト参照は、割当てられてはいるがロードされていないページに対するページ不在を生じる。ページ不在処理は実行可能オブジェクトのローディング及びアドレス再配置を生じる。共用メモリ・システムは、明示的に参照されるまで、実行可能オブジェクトの共用を許可する。



【特許請求の範囲】

【請求項1】 実行可能プログラムに対するメモリの割当てを管理するシステムであって、

(i) 1次実行可能プログラムをメモリにロードするための要求を受信する手段と、

(i i) 上記1次実行可能プログラムにより参照される2次実行可能プログラムを決定する手段と、

(i i i) 上記2次実行可能プログラムをメモリにロードすることなく、該プログラムにメモリ・スロットを割当てする手段と、

(i v) 上記割当てられてロードされていないあるスロットに対する参照を検出する手段であって、該参照が検出される時の検出信号の生成を検出する手段と、

(v) 上記検出信号にตอบสนองして、上記スロットを割当てられた実行可能プログラムをメモリにロードする手段と、を含むシステム。

【請求項2】 実行可能プログラムをロードする上記手段が、

(a) 上記実行可能プログラムを記憶装置からコピーする手段と、

(b) 上記実行可能プログラム内のアドレス参照を再配置する手段と、

(c) 上記実行可能プログラムにより参照される実行可能プログラムを決定する手段と、

(d) 上記参照される実行可能プログラムをメモリにロードすることなく、該プログラムにメモリ・スロットを割当てする手段と、

を含む、請求項1記載のシステム。

【請求項3】 上記検出手段がページ不在検出システムである、請求項2記載のシステム。

【請求項4】 上記ロード手段が外部ページャである、請求項1記載のシステム。

【請求項5】 上記ロード手段が外部ページャである、請求項3記載のシステム。

【請求項6】 実行可能ルーチンを動的にロードする方法であって、

(i) メイン実行可能ルーチンをメモリにロードするための要求を受信するステップと、

(i i) 上記メイン実行可能ルーチンにより参照される全てのルーチンのリストを作成するステップと、

(i i i) 上記ルーチンをロードすることなく、該ルーチンにメモリ・ロケーションを割当てするステップと、

(i v) ある上記割当てられたメモリ・ロケーションに対する参照を検出し、検出信号を生成するステップと、

(v) 上記検出信号にตอบสนองして、上記ルーチンを上記割当てられたメモリ・ロケーションにロードするステップと、

を含む方法。

【請求項7】 上記ロード・ステップが、

(a) 上記ルーチンを記憶装置からコピーするステップと、

(b) 上記ルーチン内のアドレス参照を再配置するステップと、

(c) 上記ロード・ルーチンにより参照されるルーチンのリストを作成するステップと、

(d) 上記参照されるルーチンをメモリにロードすることなく、該ルーチンにメモリ・ロケーションを割当てするステップと、

を含む、請求項6記載の方法。

【請求項8】 上記検出ステップがメモリ・ページ不在参照を検出するステップを含む、請求項6記載の方法。

【請求項9】 上記ロード・ステップが該ロード・ステップを実行するために、制御を外部ページャに渡すステップを含む、請求項6記載の方法。

【発明の詳細な説明】

【0001】

【産業上の利用分野】 本発明は情報処理システムに関し、特にオペレーティング・システムに関する。更に詳しくは、本発明はオペレーティング・システムにより実行可能タスクのロードを制御するシステムに関する。

【0002】

【従来の技術】 コンピュータ・オペレーティング・システムはコンピュータ・システム・コンポーネントの動作を制御する。オペレーティング・システムの役割の中に、実行用の他のプログラムのロードがある。他のプログラムは典型的にはアプリケーション・プログラムであるが、オペレーティング・システム・コンポーネントも含まれる。プログラムのロードには、プログラム記憶媒体上、典型的にはハード・ディスク・ファイル上で要求プログラムを見出し、当該プログラムをメモリ内に導き、それをプロセッサにより実行する用意をすることが含まれる。

【0003】 プログラムはプログラム・ソース・コードのコンパイルまたはアセンブルによって生成される実行形式で、プログラム記憶領域に記憶される。実行可能形式はローダによって認識される指定フォーマットを有し、実行可能コードを処理するために必要な情報を含む。ローダが実行する処理ステップの1つはアドレス再配置である。実行可能ファイル内のアドレスは開始アドレスに関連して表現される。この開始アドレスまたは開始オフセットは、プログラムがメモリ内にロードされるまで決定されない。アドレス再配置は相対アドレスを変更し、それによりこれらのアドレスはシステム内の実際のアドレス可能記憶位置をさすようになる。アドレスが指定されると、プログラムは実行の準備が整う。

【0004】 システム・ユーザ要求が拡大し、システムの能力が増すにつれ、コンピュータ・プログラムは益々複雑化しつつある。IBM PS/2コンピュータ・システム(PS/2はIBMの登録商標)などのマイクロコンピュータ

タ・システムの向上された能力が1例として挙げられる。向上されたプロセッサ能力及び機能は、複雑な文書処理、スプレッド・シート、データベース、及び他の事務及び科学アプリケーションの開発を可能としてきた。しかしながら、複雑なアプリケーションは著しく大きな実行可能ファイルを生じる。

【0005】大きな実行可能ファイルの生成は、アプリケーションまたはシステムの1部が変更される時、アプリケーションまたはシステムを作成または訂正するために要求されるコスト及び時間を増加させる。この問題に対する1つの解答が、アプリケーションまたはシステムを動的にロード可能なセグメントに分割することであった。これらのセグメントのリンク（アドレス再配置）は実行時まで延期される。各セグメントはコードの特定の部分を含むように定義され、好適に定義されたインタフェースに応答するように作成される。特定のセグメント・オブジェクトをアクセスする他のセグメント内のプログラムは、そのインタフェースが一定な限り、変更または再コンパイルされる必要がない。これはオペレーティング・システム及びアプリケーション・プログラムの開発及び保守において、より高い柔軟性を提供する。

【0006】ダイナミック・リンクの概念はIBM OS/2オペレーティング・システム製品（OS/2はIBMの登録商標）で実施された。OS/2ダイナミック・リンク・ライブラリ（またはDLL）が、オペレーティング・システムまたはアプリケーション・プログラムに定義された関数を提供するために、動的にリンクされる。DLLとのインタフェースは指定されたライブラリ・インタフェースである。ダイナミック・リンク・ライブラリを使用するプログラムは、そのプログラムに対するライブラリ・インタフェース呼出しを含む。コンパイラはこれらの外部ライブラリ呼出しの各々を追跡し、実行可能ファイルのセクションにそれらを含む。現行のシステムでは、OS/2ローダが実行可能プログラム（典型的には、EXE拡張子を有するファイルとして記憶される）をロードする時、ローダは全てのダイナミック・リンク・ライブラリ参照を見出すためにファイルを走査する。参照される各ライブラリ関数が次に動的にロードされる。ローダはこれらのライブラリの各々をチェックし、それらが他のライブラリ関数を呼出すか否かを判断する。このチェック及びロード処理は、全てのダイナミック・リンク参照が解析されるまで、再帰的に継続される。

【0007】現行のOS/2オペレーティング・システムは、タスクによって参照される各ライブラリに、メモリ・アドレス及びユーザ・メモリ空間を割当てる。共通に使用されるDLLは頻繁に他のDLLを参照し、これらは更に他のDLLを参照する。その結果、DLLの長い連鎖が割当てられ、各タスクにマップされる。これらの多くのDLLはエラーまたは他の例外状態を扱うために指定され、稀にしか使用されない。これはロードされた

大きなメモリがほとんどの時間、不要であることを意味する。大きなメモリ割当てはまたシステムのオーバヘッドを増す。なぜなら、メモリ・マネージャがページング・システム内に置換する領域を決定する時、この大きな割当てられたメモリを通じてシフトしなければならないからである。

【0008】カーネギ・メロン（Carnegie Mellon）大学により開発されたMa c hオペレーティング・システムなどのマイクロカーネル・システムでは、別の問題が存在する。Ma c hマイクロカーネルはプロセス間通信にカーネル・ポート識別子を割当てる。Ma c hカーネルはカーネル・ポート識別子を各DLL、またはより効率的に各DLLオブジェクトに割当てる。未参照のポートの割当ては効率的なポート解決を妨げる。

【0009】動的なサブルーチン・ロード技法が以前に提案されているが、各々は欠点を有した。IBM Technical Disclosure Bulletin, pp. 5535-5537（1986年5月発行）には、“Transparent Dynamic Subroutine Loader”が記載されている。ここで提案される技法は、メイン・ルーチンとリンクされる特殊目的の動的ローダを要求する。ローダはサブルーチン参照のテーブルを保持し、サブルーチンが呼ばれるとそれらをロードする。このアプローチはオペレーティング・システムにとって不可欠ではなく、タスク間の共用をサポートしない。トラッキング及びローディングが特定のタスクにのみ適用される。動的ローダ・プログラムのメイン・プログラムへのリンク要求も透過性を制限する。

【0010】多少異なるアプローチが、IBM Technical Disclosure Bulletin, pp. 3521（1987年1月発行）記載の“Execution Time Loading Using Self Modifying Stubs”に提案されている。このアプローチでは各メイン・プログラムにサブルーチン・スタブが追加されることを要求する。サブルーチンが最初に参照される時、ライブラリのコピーがロードされ、ロードされたライブラリのコピーに分岐するためにスタブが変更される。このアプローチは特殊なコーディング技法、及びプログラムのフローに対する実行時間の変更を要求する欠点を有する。

【0011】最後に、IBM Technical Disclosure Bulletin, pp. 209-210（1991年11月発行）記載の“Dynamic Link Library Mechanism in DOS Environment”では、DOS環境においてライブラリ・ルーチンのリストを管理する“terminate and stay resident”（TSR）ルーチンが提案されている。このルーチンは、メイン・ルーチンに挿入される特殊な結合呼出しにより生成される“ソフト割込み”に応答する。割込みが受信されると、TSRルーチンがテーブルをチェックし、関数が以前にロードされたか否かを判断し、否定の場合、その関数がロードされ、テーブルが更新される。制御は次に当該関数に移行する。このアプローチは動的ライブラリを管理

するために特殊な割込みルーチンを要求する欠点を有する。メイン・プログラム内に特殊な割込み呼出しを挿入する要求は、また一般のアプリケーションをも制限する。

【0012】

【発明が解決しようとする課題】提示された技術的な問題は、実行可能ライブラリ・オブジェクトが実行要求される時にだけ、これらのオブジェクトを動的にロードする機構を提供することである。この問題の解決は、アプリケーション開発者及び実行プログラムにとって透過的となるべきである。すなわち、ライブラリ関数がメイン・プログラムにリンクされる場合と差異ないように見えないなければならない。

【0013】

【課題を解決するための手段】本発明はライブラリ・オブジェクトが実行のために要求される時にだけ、それらを動的にロードするシステムを目的とする。このシステムは未参照のオブジェクトのロードを、参照が行われるまで遅延させる手段を含む。本発明はこの機能をシステム・ローダ及び実行処理の1部として組み込み、メイン実行可能プログラムの変更は要求しない。

【0014】本発明はメモリの実行可能プログラムへの割当てを管理するシステムを含み、このシステムは、1次実行可能プログラムをメモリにロードするための要求を受信する手段、1次実行可能プログラムにより参照される2次実行可能プログラムを決定する手段、プログラムをメモリにロードすることなく、メモリ・スロット（領域）を2次実行可能プログラムに割当てする手段、割当てられロードされていないあるスロットへの参照を検出する手段、参照が検出される時の検出信号の生成を検出する手段、検出信号に応答して、スロットを割当てられた実行可能プログラムをメモリにロードする手段を含む。

【0015】本発明の上述の目的、機能、及び利点が、添付の図面により表される後述の実施例により明らかにされ、ここで同一参照番号は本発明の同一のパーツを表す。

【0016】

【実施例】本発明は情報処理システムに関連して使用される。こうしたシステムの例が図1の100に示される。このシステムは処理ユニットまたはCPU102を有し、これはシステム・バス103により様々なシステム・コンポーネントに相互接続される。これらのコンポーネントにはハード・ディスク記憶装置104、ランダム・アクセス・メモリ（RAM）106、ネットワーク・インタフェース108、入出力インタフェース110などが含まれるが、これらに限るものではない。入出力インタフェースはキーボード114、ポインティング装置116からの入力を調整し、表示装置112への出力を制御する。ディスク・ドライブ、テープ・ドライ

ブまたはCD-ROMドライブなどの永久記憶装置を更にこうしたシステムに接続することができる。

【0017】現行のシステムが動作する情報処理システムは、好適にはIBM PS/2コンピュータ・システムである。しかしながら、ここで開示されるシステム及び方法はこうしたシステムに限るものではなく、他のマイクロコンピュータ・システム、IBMRISC System/6000ワークステーション（RISC System/6000はIBMの登録商標）などのワークステーション・システム、またはIBM ES/9000 コンピュータ・システム（ES/9000はIBMの登録商標）上においても動作可能である。

【0018】本発明の実施例はダイナミック・リンク・ライブラリ（DLL）をサポートするシステムにおいて実施される。これらのライブラリは、ライブラリ内の各オブジェクトがロード時に再配置可能である、すなわち各オブジェクトがローダにより任意のアドレスに割当てられる特性を有する。

【0019】各実行可能モジュールは永久記憶装置（例えばハード・ディスク上）に、指定されるフォーマットで記憶される。実施例ではメイン実行可能ファイル及び全てのダイナミック・リンク・ライブラリが、“LX（Linear Executable Module）”フォーマットに従い記憶される。記憶フォーマットは全てのフォーマットがローダにより認識可能である限り、メイン実行及びダイナミック・リンク・ライブラリの間で、またはダイナミック・リンク・ライブラリ内においても異なることが可能である。

【0020】実行可能モジュールの例が図2の202に示される。メイン実行可能モジュールは、関数を実行するためにオペレーティング・システムまたは直接ユーザにより呼出されるプログラムである。ユーザ呼出しの例は、オペレーティング・システム・コマンド・プロンプト位置における、ユーザによるプログラム名の入力である。例えば、MYPROGMの入力により、MYPROGM. EXEがロードされて実行される。

【0021】実行可能モジュール・ヘッダは、プログラムをロード及び実行するために要求される様々な情報を含む。これらにはモジュール名、バージョン・レベル、CPUまたはオペレーティング・システム依存性、モジュールのサイズ、アドレス再配置のために必要なアドレス再配置（fixup）情報、及び実行に際し搬入されるDLLモジュールに対する搬入モジュール参照が含まれる。

【0022】メイン実行可能プログラムはインポート・モジュール・リストを通じて、ダイナミック・リンク・ライブラリを参照する。DLL参照はシステム開発者により挿入される明示的なプログラム呼出しのために発生するか、またはコンパイルの間にコンパイラにより挿入される。図2に示されるように、モジュール202はDLL204及び206内のオブジェクトに対する参照を

含む。モジュール204はモジュール208、210及び212に対する参照を含み、モジュール206はモジュール212及び214に対する参照を含む。

【0023】各ダイナミック・リンク・ライブラリ(DLL)は、テキスト、共用データ、及びインスタンス・データを含む多数のオブジェクトに分割される。実行可能オブジェクトは、オブジェクトの生成時に選択されるオプションに依存して、いくつかのまたは単一のルーチンを含む。各オブジェクトはロード時に再配置される。すなわち、ロード時に任意のアドレスに割当てられる。各オブジェクトは更に、コンピュータ・システムの仮想メモリ・マネージャ内で使用されるページのサイズに対応して、ページに分割される。各ページは関連する“アドレス再配置ストリーム(fixup stream)”を有し、これはモジュールのロード時に変更されなければならないアドレスを識別し符号化する。これらの変更は局所アドレス再配置と外部アドレス再配置に分かれ、前者では局所アドレスがオブジェクトの開始アドレスによりオフセットされ、後者ではアドレスが他のオブジェクトまたは他のダイナミック・リンク・ライブラリのアドレスにもとづきオフセットされる。

【0024】本発明の実施例はOS/2オペレーティング・システムの性質を有するMachマイクロカーネル基本システムを使用して実施される。このシステムの主な機能コンポーネントが図3に示される。ユーザ302はユーザ空間で実行されるプログラムである。これはアプリケーションまたはシステム・ユーティリティ・プログラムである。OS/2サーバ304はOS/2のオペレーティング・システム機構を含み、これはOS/2ユーザ・インタフェースを提供し、マイクロカーネル306に対して必要なマイクロカーネル・サービス要求を生成する。ファイル・サーバ308はディスク・ファイルまたは他の記憶媒体を管理する。

【0025】OS/2サーバ304は本発明に関わる3つのコンポーネントを含む。タスキング310は要求動作を実行するための実行タスクを確立する。ローダ312はファイル・サーバ308から実行可能モジュールを要求するための関数を実行し、実行可能モジュールを取り決め、それらをメモリ内にロードする。メモリ・マネージャ314はマイクロカーネル・メモリ管理機構を使用して、メモリ資源を管理する。マイクロカーネルは仮想(VM)メモリ・システム316を有する。仮想メモリ・システムはVMページのメモリ・オブジェクトへの割当てを管理し、メモリ・ページ不在を検出する。ページ不在はタスクに割当てられているが、現在メモリ内に存在しないメモリ・ページを要求するタスクにより起こされる。不在の扱いとしては、未使用または最近使用されていないページをページ・アウトし、要求ページをページ・インすることが要求される。

【0026】本発明の動的ローディング処理の動作につ

いて、図3及び図4を参照しながら説明する。処理は実行可能プログラム初期化のためのユーザ空間要求により開始される(402)。これはタスキング310に実行されるタスクを生成させ、そのタスクを仮想メモリ・システム316に登録させる。タスキングは次にローダ312に、EXEファイルのメモリ・ロードを開始させる(404)。ローダはファイル・サーバ308を通じて、EXEファイルをアクセスする。

【0027】ローダは初期実行可能コードのロードを開始する(406)。ローダは、EXEヘッダをフェッチし、モジュールに対応するモジュール・テーブル・エントリ(MTE)を生成する。モジュール・テーブル・エントリがVM316に登録される。ローダは、EXEファイルにより参照されるDLLのリストを作成する。各DLLオブジェクトに対し、ローダはMTEがそのDLLオブジェクトに対応して存在するか否かを判断し、否定の場合、ファイル・サーバを用いてDLLをアクセスし、DLLオブジェクトに対応するMTEをロードし生成する。MTEはVMに登録される。

【0028】タスクの実行が次に開始される(408)。ローディング処理は、EXE及び参照されるDLLに関連するデータの各ページをユーザの仮想メモリ空間にマップする。しかし、物理ページは参照されるまでロードされない。ロードされない仮想メモリ・ページの参照はページ不在を引起こす(410)。メモリ・オブジェクト・ページ不在を処理するために外部ページャ(pager)が提供される。このページャはユーザ・メモリに要求ページをロードする。ページが1度ロードされると、デフォルトのマイクロカーネル・ページャが続くローディングまたはアンローディングを処理する。

【0029】ページ不在の場合(410)、マイクロカーネルVM316はオブジェクト名、ページ、及びMTEを決定する。有効ページに対し、外部ページャはローダにページをアドレス再配置させ、ロードするように命じる(416及び418)。アドレス再配置処理は任意の別の外部オブジェクトすなわち他のDLLオブジェクトを検出し、そのオブジェクトがメモリを割当てられているか否かを判断する(420)。否定の場合、適切なサイズのメモリ・スロットが要求され(422)、ページがユーザ・アドレス空間にマップされる(424)。

【0030】実施例はタスクが特定のメモリ領域を共有可能な共用メモリ・システムにおいて実施される。別のユーザ・タスクがあるユーザに既に割当てられているページを参照する時(414)、カーネルはその第2のユーザ・タスクにカーネル・キャッシュからそのページのコピーを提供し、ローダがページのアドレス再配置を行う必要性を回避する。この共用は、第2のユーザのために再度そのページの読出し及びアドレス再配置を行うオーバーヘッドを回避する。第2のユーザが共用ページ上のオブジェクトを参照する時、メモリ保護不在が発生す

る。アドレス再配置が成されていないために、共用ページからのオブジェクトはユーザ空間にはロードされず、固定ページのコピーだけが実行される。サーバはオブジェクトを第2のユーザのアドレス空間にマップすることにより不在を処理し(430)、ユーザ・タスクを再スタートする。

【0031】上述の処理はライブラリ・オブジェクト要求の評価(evaluation)を遅延させるか、レイジ(lazy)にする。ローダはライブラリ・オブジェクトが使用されようとするまで、そのロードを延期する。EXEファイル内の全てのDLLオブジェクト参照は初期にロードされる。なぜならば、初期実行がこれらのオブジェクトのいずれをも含むことができるからである。各DLLオブジェクトが参照される時、そのオブジェクトにより参照される次のレベルのDLLオブジェクトが処理される。

【0032】以下のように発明を開示する。

(1) 実行可能プログラムに対するメモリの割当てを管理するシステムであって、(i) 1次実行可能プログラムをメモリにロードするための要求を受信する手段と、(i i) 上記1次実行可能プログラムにより参照される2次実行可能プログラムを決定する手段と、(i i i) 上記2次実行可能プログラムをメモリにロードすることなく、該プログラムにメモリ・スロットを割当てする手段と、(i v) 上記割当てられてロードされていないあるスロットに対する参照を検出する手段であって、該参照が検出される時の検出信号の生成を検出する手段と、(v) 上記検出信号に応答して、上記スロットを割当てられた実行可能プログラムをメモリにロードする手段と、を含むシステム。

(2) 実行可能プログラムをロードする上記手段が、

(a) 上記実行可能プログラムを記憶装置からコピーする手段と、(b) 上記実行可能プログラム内のアドレス参照を再配置する手段と、(c) 上記実行可能プログラムにより参照される実行可能プログラムを決定する手段と、(d) 上記参照される実行可能プログラムをメモリにロードすることなく、該プログラムにメモリ・スロットを割当てする手段と、を含む、(1)記載のシステム。

(3) 上記検出手段がページ不在検出システムである、

(2)記載のシステム。

(4) 上記ロード手段が外部ページである、(1)記載のシステム。

(5) 上記ロード手段が外部ページである、(3)記載のシステム。

【0033】(6) 実行可能ルーチンを動的にロードする方法であって、(i) メイン実行可能ルーチンをメモリにロードするための要求を受信するステップと、(i i) 上記メイン実行可能ルーチンにより参照される全てのルーチンのリストを作成するステップと、(i i i) 上記ルーチンをロードすることなく、該ルーチンにメモリ・ロケーションを割当てするステップと、(i v) ある

上記割当てられたメモリ・ロケーションに対する参照を検出し、検出信号を生成するステップと、(v) 上記検出信号に応答して、上記ルーチンを上記割当てられたメモリ・ロケーションにロードするステップと、を含む方法。

(7) 上記ロード・ステップが、(a) 上記ルーチンを記憶装置からコピーするステップと、(b) 上記ルーチン内のアドレス参照を再配置するステップと、(c) 上記ロード・ルーチンにより参照されるルーチンのリストを作成するステップと、(d) 上記参照されるルーチンをメモリにロードすることなく、該ルーチンにメモリ・ロケーションを割当てするステップと、を含む、(6)記載の方法。

(8) 上記検出ステップがメモリ・ページ不在参照を検出するステップを含む、(6)記載の方法。

(9) 上記ロード・ステップが該ロード・ステップを実行するために、制御を外部ページに渡すステップを含む、(6)記載の方法。

【0034】(10) 実行可能ルーチンをメモリに動的にロードするための、コンピュータ・プログラム論理が記録されるコンピュータ読出し可能媒体を有するコンピュータ・システムであって、(i) 1次実行可能ルーチンをメモリにロードするための要求を受信する手段と、(i i) 上記1次実行可能ルーチンにより参照される2次実行可能ルーチンを決定する手段と、(i i i) 上記2次実行可能ルーチンをメモリにロードすることなく、該ルーチンにメモリ・スロットを割当てする手段と、(i v) 上記割当てられてロードされていないあるスロットに対する参照を検出する手段であって、該参照が検出される時の検出信号の生成を検出する手段と、(v) 上記検出信号に応答して、上記スロットを割当てられた上記実行可能ルーチンをメモリにロードする手段と、を含むコンピュータ・システム。

(11) 実行可能ルーチンをロードする上記手段が、

(a) 上記実行可能ルーチンを記憶装置からコピーする手段と、(b) 上記実行可能ルーチン内のアドレス参照を再配置する手段と、(c) 上記実行可能ルーチンにより参照される実行可能ルーチンを決定する手段と、

(d) 上記参照される実行可能ルーチンをメモリにロードすることなく、該ルーチンにメモリ・スロットを割当てする手段と、を含む、(10)記載のコンピュータ・システム。

(12) 上記検出手段がページ不在検出システムである、(11)記載のコンピュータ・システム。

(13) 上記ロード手段が外部ページである、(10)記載のコンピュータ・システム。

(14) 上記ロード手段が外部ページである、(12)記載のコンピュータ・システム。

【0035】

【発明の効果】以上説明したように、本発明はメモリ割

当てを制限し、参照オブジェクトにより要求されるメモリ割当てにロードする利点を有する。これはまた、割当てられなければならないMac hポート及びメモリ・ページの数を制限し、それにより全体的なシステムのオーバーヘッドを低減する。

【0036】実施例はMac hマイクロカーネルに関連して示された。しかしながら、当業者には理解されるように、レイジ・ローディング処理はこのタイプのオペレーティング・システムに限るものではない。ここで述べられる処理は他のオペレーティング・システム、特に仮想メモリ管理をサポートするシステムに適用可能である。

【図面の簡単な説明】

【図1】 本発明で使用する情報処理システムを表すブロック図である。

【図2】 実行可能オブジェクトの構造及び相関を示す図

である。

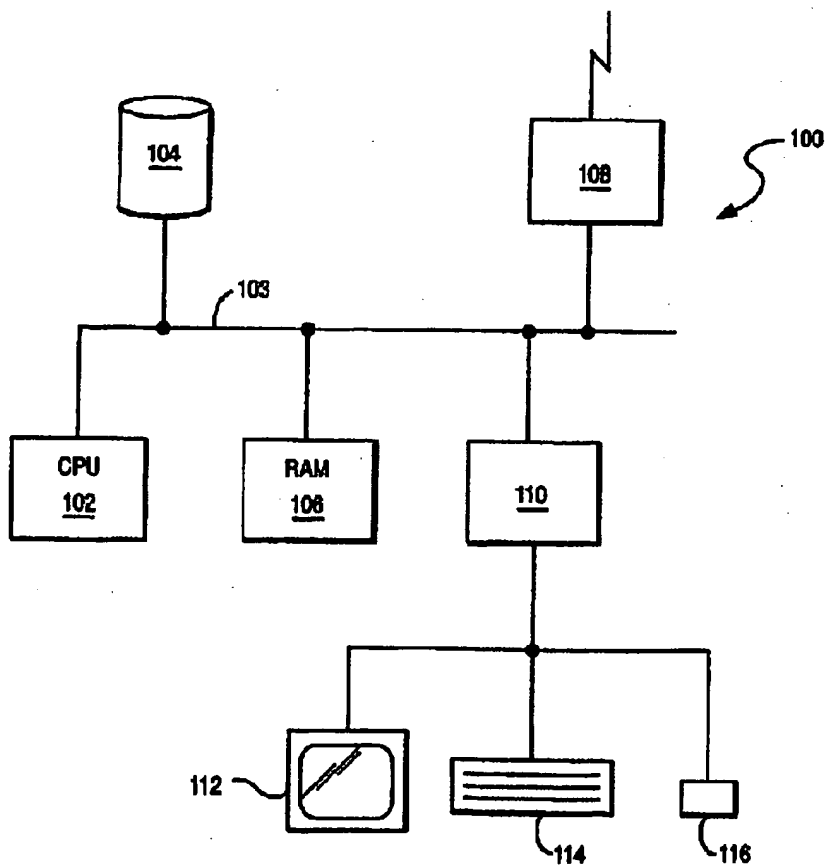
【図3】 本発明の実施例のコンポーネントを示すブロック図である。

【図4】 本発明の処理ステップを表す流れ図である。

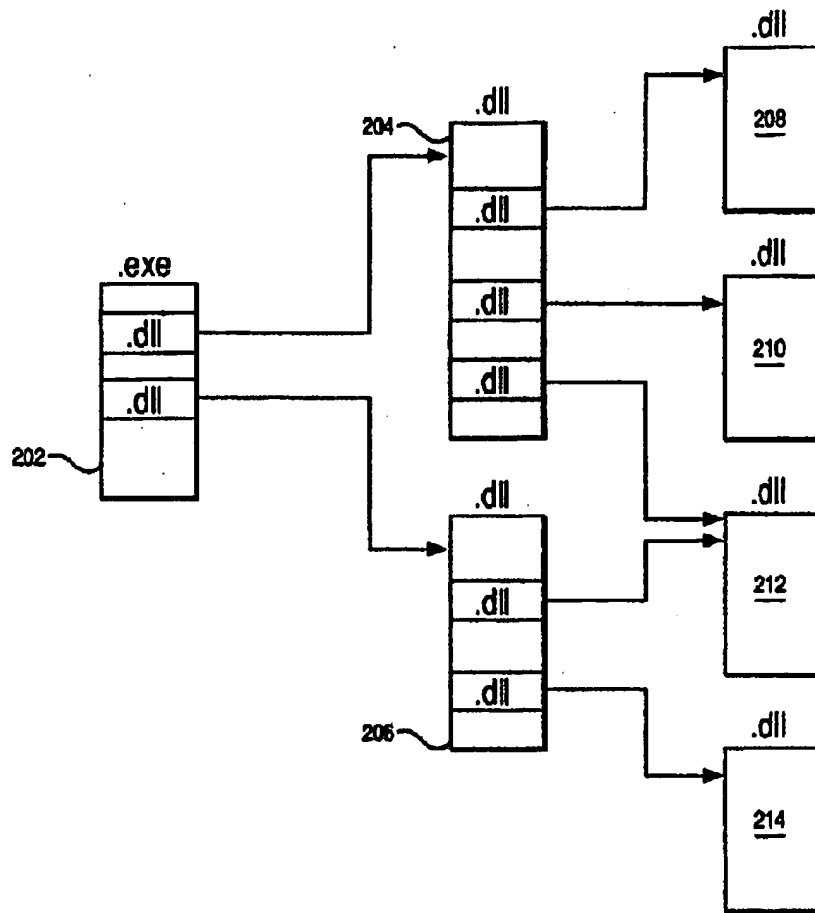
【符号の説明】

- 102 CPU
- 104 ハード・ディスク記憶装置
- 106 ランダム・アクセス・メモリ (RAM)
- 204、206 DLL
- 302 ユーザ
- 304 OS/2サーバ
- 308 ファイル・サーバ
- 310 タスキング
- 312 ロード
- 314 メモリ・マネージャ
- 316 仮想メモリ・システム (VM)

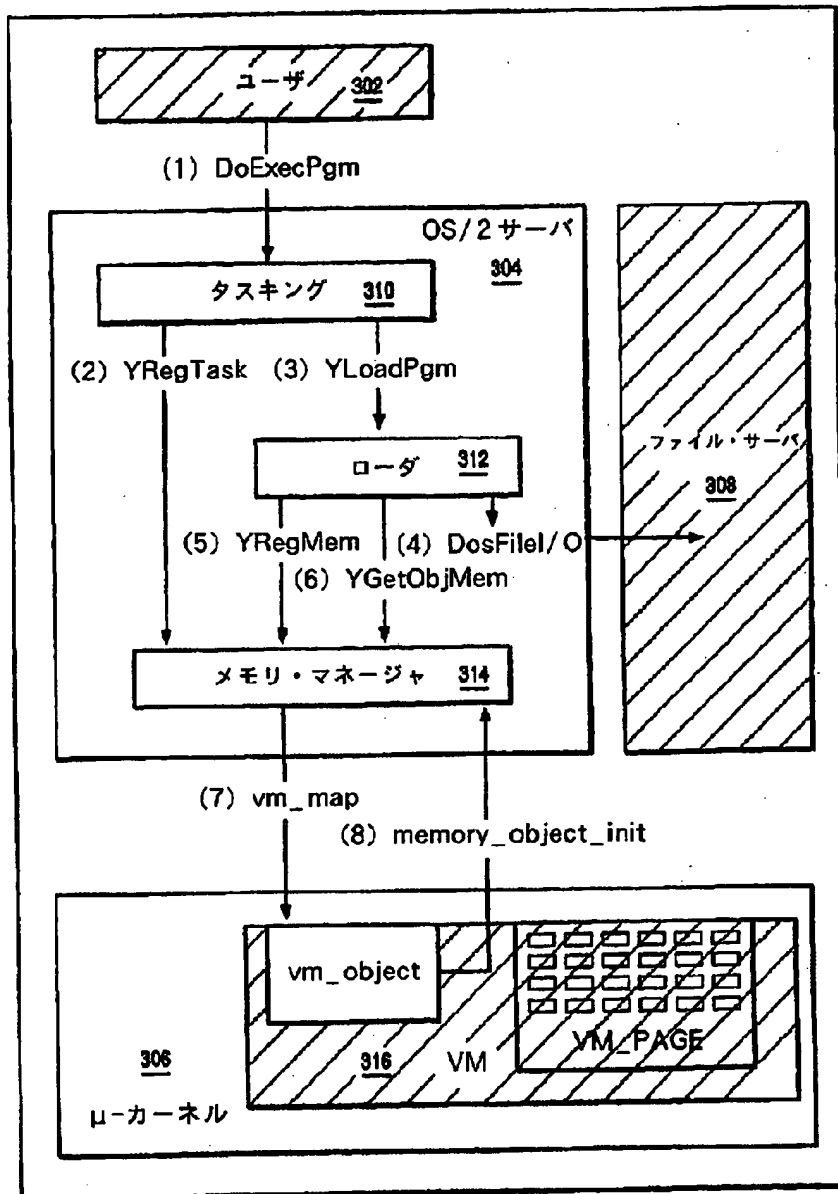
【図1】



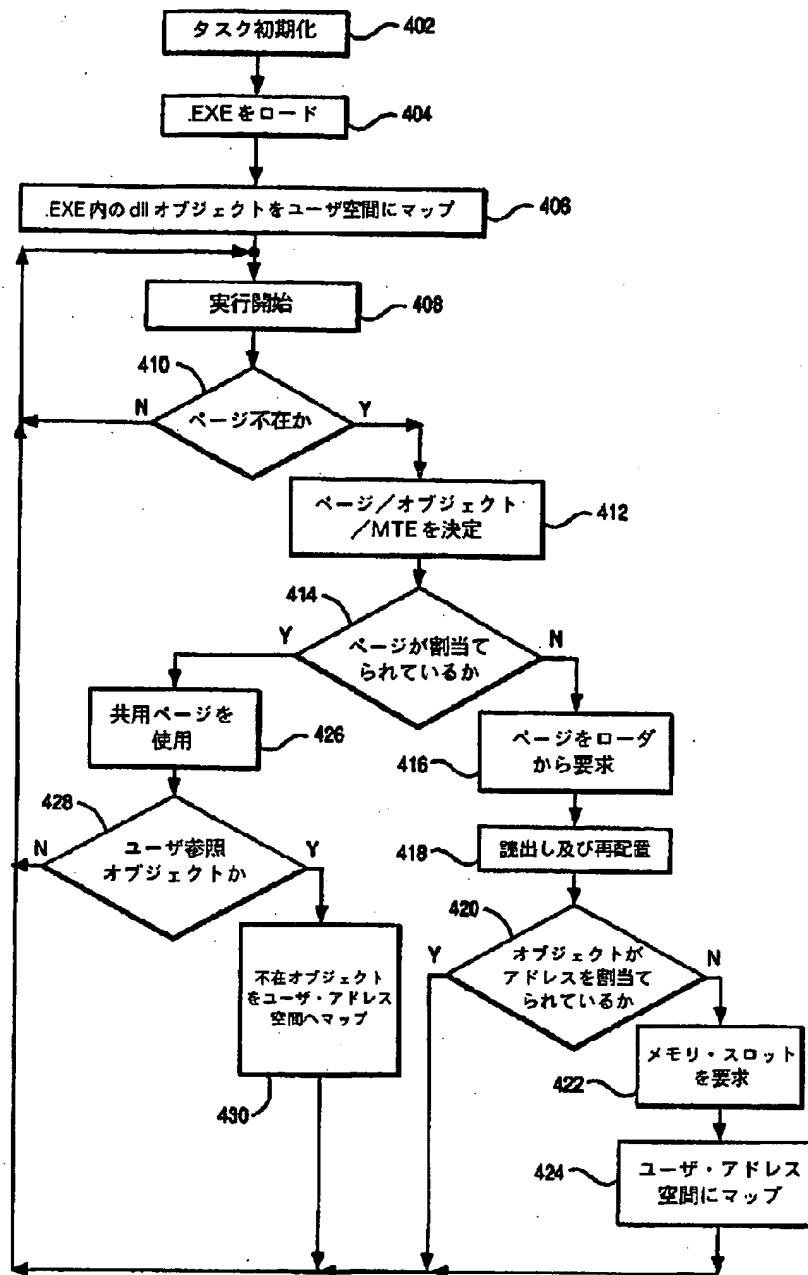
【图 2】



【図3】



【図 4】



フロントページの続き

(72)発明者 ポール・ブラシド・ジアンガーラ
アメリカ合衆国33434、フロリダ州ボカ
ラトン、ノース・ウエスト・フォーティ
ス・ストリート 2591

(72)発明者 ラビンドラナス・カシナス・マニクンダ
ム
アメリカ合衆国78728、テキサス州オース
ティン、ウッド・チェイス・トレイル
2406

(72)発明者 ドナルド・ロバート・パジェット
アメリカ合衆国78731、テキサス州オース
ティン、キャット・クリーク・トレイル
6604

(72)発明者 ジェームス・マイケル・ペラン
アメリカ合衆国78731、テキサス州オース
ティン、ナンバー136、ドライ・クリー
ク・ドライブ 3839